Politechnika Wrocławska

Sztuczna inteligencja i inżynieria wiedzy

Laboratorium - Lista 2

9.05.2024

Krzysztof Głowacz, 266545

Spis treści

1 Wstęp

Gra planszowa **Halma** jest przykładem gry o sumie zerowej. W trakcie wykonywania niniejszej listy zadań rozważany był przypadek gry dwóch graczy, z których każdy dysponuje dziewiętnastoma pionkami, których celem jest przemieszczenie się z jednego narożnika planszy do przeciwległego. Zgodnie z zasadami gry [3][1] w każdej turze gracz musi wykonać jeden ruch, który może być:

- przesunięciem pionka na dowolne, niezajęte przez innego pionka pole bezpośrednio sąsiadujące z polem, na którym aktualnie stoi pionek (licząc także pola stykające się narożnikami),
- przeskoczeniem pionka, przy czym skok można wykonać jedynie nad innym pionem (swoim lub przeciwnika), gdy pole bezpośrednio za polem zajmowanym przez przeskakiwanego pionka (w linii prostej) jest wolne. W trakcie ruchu można wykonać dowolnie wiele takich skoków, jeśli następują one kolejno po sobie, tzn. gdy pole końca jednego skoku jest jednocześnie polem startu następnego. Nie ma konieczności wykonywania wszystkich możliwych skoków - można taki manewr przerwać w każdym momencie.

Gra kończy się w momencie, gdy jeden z graczy przemieści wszystkie swoje pionki do bazy przeciwnika będącej w naprzeciwległym narożniku planszy w stosunku do tego, w którym gracz rozpoczyna grę.

2 Teoria

2.1 Opis teoretyczny metody

Program grający w grę Halma oparty jest o algorytm Minimax. Przed wykonaniem każdego ruchu budowane jest drzewo decyzyjne, w którym węzeł symbolizuje stan planszy, a jego dzieci są możliwymi następnymi stanami planszy po wykonaniu pojedynczego ruchu przez jednego z graczy. W ten sposób „gracz" jest w stanie dobierać swoje ruchy planując je z wyprzedzeniem, starając się uwzględniać także możliwe odpowiedzi rywala. Ze względu na wysoki współczynnik rozgałęzienia (spowodowany dużą liczbą możliwych ruchów w każdej turze) obliczeniowo niemożliwe jest osiągnięcie dużej głębokości dla standardowej maszyny. Podczas testów i przeprowadzanych eksperymentów maksymalna stosowana głębokość drzewa była równa 3. Do oceny wartości liści stosowane były różnego rodzaju heurystyki, które szerzej opisane zostały w następnym rozdziale.

2.2 Formalne sformułowanie problemu

Aby formalnie opisać rozgrywkę jako problem z ograniczeniami (*Constraint Satisfaction Problem*) należy zdefiniować zbiór zmiennych, ich dziedzin oraz zbiór ograniczeń wiążących te zmienne [2].
Niech:

$\mathcal{X} = \{X_{(0,1,t)}, X_{(1,1,t)}, \ldots, X_{(15,1,t)}, X_{(0,2,t)}, \ldots X_{(15,2,t)}\}$ - zbiór zmiennych, gdzie $X_{(i,k,t)}$ oznacza i-ty pionek k-tego gracza w turze t. Wartością zmiennej jest para (x,y), która oznacza umiejscowienie danego pionka na przecięciu rzędu x i kolumny y.

$\mathcal{D} = D_{0,1} \times D_{1,1} \times \ldots \times D_{15,1} \times D_{0,2} \times \ldots \times D_{15,2}$ - swobodna przestrzeń przeszukiwań, przy czym $D_{i,k}$ jest dziedziną i-tej zmiennej k-tego gracza. Dziedziny te są sobie równe, tzn. $\mathcal{D} = D^{38}$, przy czym $D = \{(0,0), (0,1), (0,2), \ldots, (0,15), (1,0), \ldots, (15,15)\}$.

$\mathcal{C} = \{C_1, C_2, C_3\}$ - zbiór ograniczeń o sygnaturze

$$C_k : \mathcal{D} \to \{prawda, fasz\},$$

gdzie:

C_1 - ograniczenie na ruch w obrębie planszy:

$$\forall_{i \in \{0,1,\ldots,15\}}, \forall_{k \in \{1,2\}}, \forall_{t \in \{1,2,3,\ldots\}} X_{i,k,t} \in D$$

C_2 - ograniczenie, że w danej turze rusza się tylko jeden pionek (ograniczenie nałożone na indeks t),

C_3 - ograniczenie, że ruch (zmiana jednej ze zmiennych) odbywa się zgodnie z regułami gry.

3 Strategie i heurystyki

Zgodnie z różnego rodzaju poradnikami do gry [4][1] strategie zostały podzielone na 3 kategorie: grę początkową, grę środkową oraz grę końcową. Do każdej kategorii zostały stworzone 2 strategie, których cele nieco się różniły. Każda strategia implementowała wspólny interfejs złożony z trzech metod: `evaluate`, `switch_strategy_check` oraz `prepare_nodes_order`. Kolejno odpowiedzialne były one za: ocenę jakości stanu planszy zgodnie z daną strategią, sprawdzenie, czy cele strategii zostały osiągnięte i należy już ją zmienić oraz za przygotowanie kolejności węzłów do rozwinięcia na kolejnym poziomie. Ostatnia z metod wykorzystywana była przez cięcia alfa-beta, o czym wspomniano szerzej w rozdziale o eksperymentach. Strategie te zostały przygotowane pod obu graczy, tzn. każdy gracz mógł użyć każdej strategii ze swojej perspektywy. Przy perspektywie gracza maksymalizującego zwracano wyniki ujemne, które opisywały np. odległość od arbitralnie zdefiniowanego punktu czy obszaru. Wówczas gracz maksymalizujący dążył do maksymalnej wartości, tzn. tej najbliższej zeru, przez co minimalizował stojącą za tą wartością odległość. Podczas symulowania ruchów gracza minimalizującego wedle danej strategii wybierane były z kolei wartości jak najmniejsze, to jest te, które odpowiadały za jak największe odległości. W ten sposób gracz minimalizujący starał się możliwie najbardziej przeszkodzić graczowi maksymalizującemu. Analogiczna sytuacja zachodziła w odwrotnej sytuacji.

3.1 Gra początkowa

3.1.1 Szybka walka o centrum

Pierwsza ze strategii do gry początkowej zakładała jak najszybsze przemieszczenie pionków w stronę centrum planszy. Motywacją gracza w tym przypadku jest fakt, że przewaga pionków w centrum może być istotnym atutem w dalszej fazie rozgrywki. Zdefiniowane zostało pole oznaczające nowe centrum grawitacyjne dla pionków danego gracza. Po rozpoczęciu gry dążyły one za wszelką cenę do zminimalizowania łącznej odległości wszystkich pionków od tego punktu. Symbolicznie strategię przedstawia zdjęcie 1(a), gdzie morski kolor pola oznacza docelowe pole strategii, wokół którego gromadziły się pozostałe piony (pola żółte).

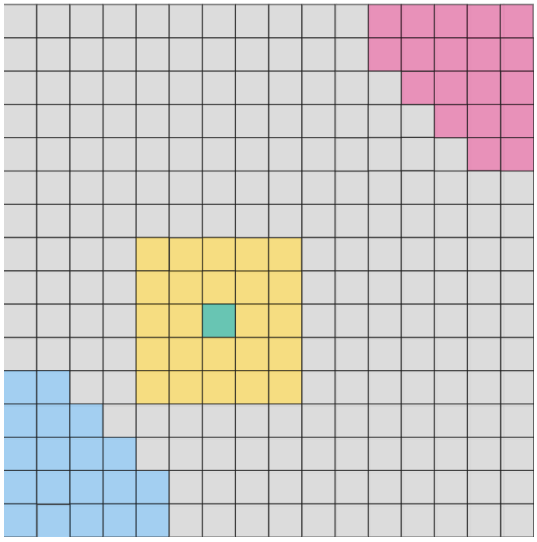
3.1.2 Walka o centrum wraz z pułapką na przeciwnika

Druga strategia dla gry początkowej była pewnego rodzaju modyfikacją tej pierwszej polegającą na tym, że gracz po przegrupowaniu pionów w okolice zdefiniowanego punktu zostawiał kilka ze swoich pionów na wyznaczonych polach tworząc przeszkodę dla pionów rywala. Przeszkoda ta znajdowała się na głównej przekątnej planszy nieprzypadkowo - jej celem było zmuszenie przeciwnika do rozproszenia swoich pionów (idących najpewniej główną przekątną) i okrążenia przeszkody (celowo przybiera ona taki kształt, aby uniemożliwić rywalowi dokonanie skoków). Aby umożliwić graczowi utrzymanie tej przeszkody przez całą grę środkową (tylko wtedy jej postawienie ma sens) skorzystano z podejścia wzorowanego na pierwszej liście laboratoryjnej - zdefiniowano zbiór tabu, z którego gracz nie mógł ruszać pionów aż do rozpoczęcia końcowej fazy gry. Wizualnie pole to przedstawione jest fioletową barwą na zdjęciu 1(b). W praktyce utrzymywanie tej pozycji przez gracza przedstawia zdjęcie 2(b), na którym widać, że mimo 52. tury jeden z graczy wciąż utrzymuje kilka swoich pionów w zwartym szyku, co zmusza rywala do obchodzenia utworzonej w ten sposób blokady.

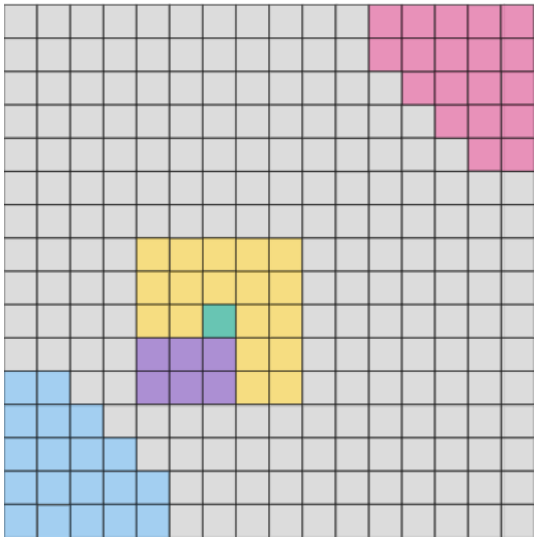
3.2 Gra środkowa

3.2.1 Przemarsz pionów główną przekątną

Pierwsza ze strategii dla gry środkowej miała na celu przeprowadzenie pionów wzdłuż głównej przekątnej do obszaru docelowego, którym był narożnik planszy zawierający docelową bazę przeciwnika. Zakładała ona, że głównym celem gracza w trakcie gry środkowej powinno być „przerzucenie" pionów najkrótszą drogą (wzdłuż przekątnej) w pobliże obszaru docelowego, ale w taki sposób, by jednocześnie nie zacząć zajmować pól będących w bazie rywala, ponieważ kolejność ich zajmowania może mieć istotne znaczenie, więc do tego służy osobna strategia dla gry końcowej. Kod realizujący tę strategię został odpowiednio przygotowany tak, by gracz nie widział żadnej korzyści z pchania piona dalej niż tylko do granic określonego obszaru. Rysunek poglądowy tej strategii przedstawia zdjęcie 2 - kolorem zielonym oznaczono dopuszczalne pola do wykonania ruchu. Im dalej od głównej przekątnej, tzn. im jaśniejszy kolor, tym większa kara. Poza obszarem głównej przekątnej naliczana kara zdecydowanie zwiększa się. Żółty obszar ograniczony czerwonymi odcinkami to docelowe pole dla tej strategii. Atutem tego rodzaju przemarszu



(a) Szybka walka o centrum - obszar docelowy



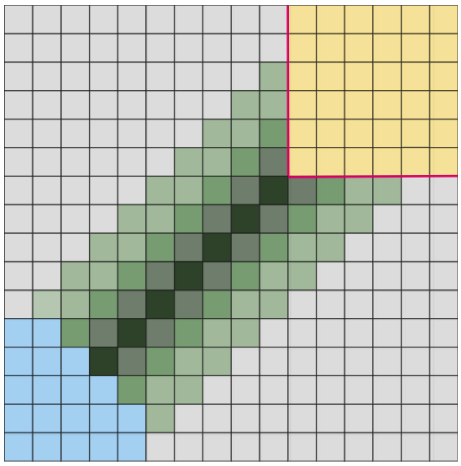
(b) Utworzenie przeszkody - rozmieszczenie pól

Zdjęcie 1: Porównanie wizualne stratgii początkowych

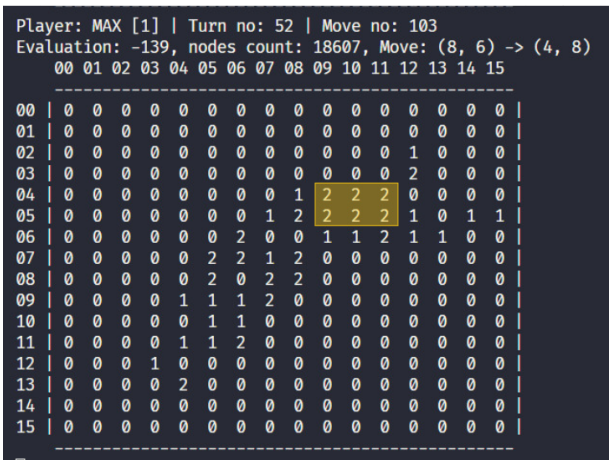
pionów jest fakt, że ograniczenie w postaci przekątnej utrzymuje piony gracza relatywnie blisko siebie, co może zwiększać liczbę możliwych skoków.

3.2.2 Rozważne stawianie pionów

Druga ze strategii także opiera się na „przerzuceniu" pionów do zbioru pól będącego nadzbiorem docelowym. Tym razem jednak nie jest istotnie przemieszanie się wzdłuż głównej przekątnej, a kontrolowanie pojedynczych ruchów tak, by nie tylko zbliżały one kolejne piony do obszaru docelowego, ale także by starały się zwiększyć liczbę możliwych skoków dla swoich sojuszników jednocześnie próbując zmniejszyć liczbę możliwych skoków dla przeciwnika. Odbywa się do w ten sposób, że przy obliczaniu jakości danego stanu planszy dla każdego piona oprócz obliczenia jego odległości od celu sprawdzane jest także, czy w danym ustawieniu pion ten „podkłada się" jakiemuś wrogowi, a także czy pion ten tworzy jakieś szanse skoków dla innych pionów gracza. Plusem tej strategii może być fakt, że bywa ona czasem bardziej elastyczna i nie zmusza pionów gracza do podążania konkretną ścieżką. Minusem natomiast jest przede wszystkim jej złożoność czasowa, szczególnie widoczna przy dokonywaniu sprawdzenia sąsiadów pionka.



(a) Strategia przemarszu wzdłuż głównej przekątnej



(b) Egzekwowanie przeszkody w praktyce

Zdjęcie 2: Strategia nr 3 oraz fragment rozegranej partii

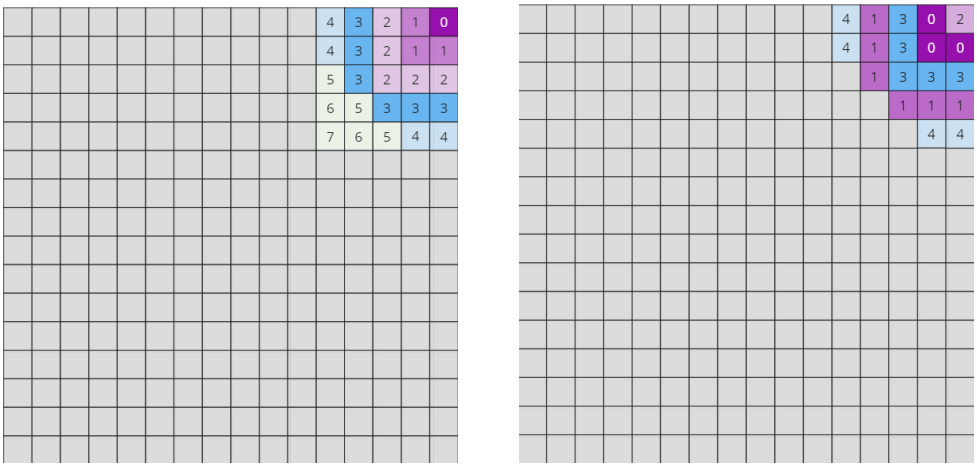
3.3 Gra końcowa

3.3.1 Wypełnianie bazy od końca

Pierwszą strategią dla gry końcowej jest strategia oparta o wypełnianie docelowego obszaru - bazy przeciwnika, od końca (narożnika) do granic. W ten sposób unikamy sytuacji, gdy piony mogłyby blokować się wzajemnie podczas próby zajęcia bazy. Do wyegzekwowania tego porządku zdefiniowane zostały odpowiednie wagi pól, które służą w obliczeniach bardziej jako kara za dystans. Te pola, które powinny być wypełnione na początku mają wagę 0, natomiast im wyższa waga, tym niższy priorytet (kolejność) danego pola. Obrazkowo przypisane w ten sposób wagi przedstawia zdjęcie 3(a).

3.3.2 Wypełnianie bazy co drugi rząd

Inna strategia dla gry końcowej naprowadza gracza na wypełnianie bazy rywala w innej kolejności - pionki zajmują najpierw co drugi rząd w nadziei, że w ten sposób przygotują miejsce ostatnim przychodzącym pionom, które w łatwy sposób za pomocą skoków będą mogły wypełnić brakujące miejsca. Wagi przypisane poszczególnym polom prezentuje zdjęcie 3(b).



(a) Wagi pól dla strategii „od końca”

(b) Wagi pól dla strategii „co drugi rząd”

Zdjęcie 3: Wagi pól dla obu strategii końcowych

4 Eksperymenty

Algorytm Minimax można przyspieszyć poprzez wprowadzenie tzw. cięcia alfa-beta (*alpha-beta prunig*). Pozwala ono wyeliminować rozpatrywanie niepotrzebnych węzłów, przy co do których jesteśmy pewni znacznie wcześniej niż na poziomie liści, że nie prowadzą one do rozwiązania optymalnego. Należy podkreślić w tym przypadku, że ta modyfikacja zupełnie nie wpływa na jakość znalezionego rozwiązania. Jedynym celem jest w tym przypadku poprawa wydajności, dzięki której możliwe jest zwiększenie maksymalnej głębokości przeszukiwanego drzewa. Wspomniana poprawa wydajności nie jest jednak gwarantowana, ponieważ silnie zależy ona od porządku rozwijania dzieci kolejnych węzłów.

Rozpatrzmy alfa-beta cięcia jako usprawnienie algorytmu Minimax. Niech:

b - średni współczynnik rozgałęzienia,

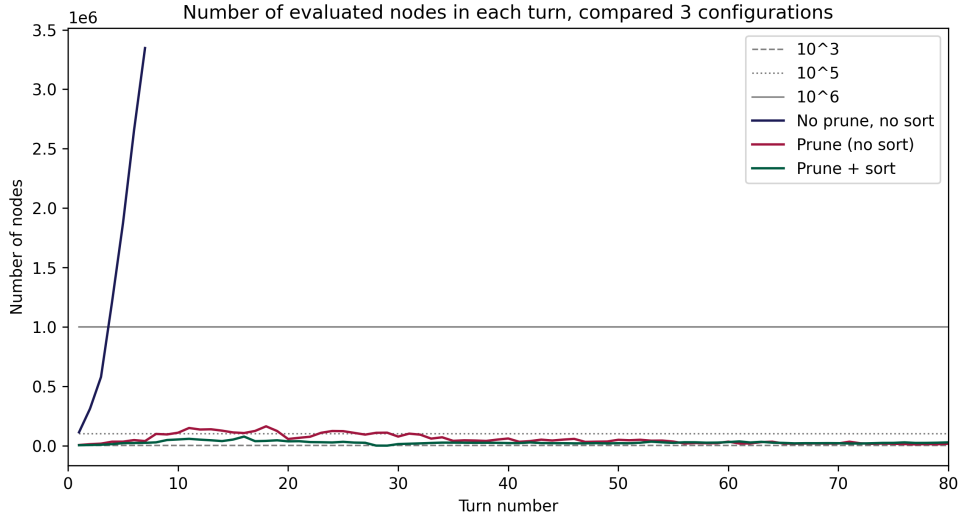
g - głębokość.

Wówczas maksymalna liczba rozpatrzonych węzłów to:

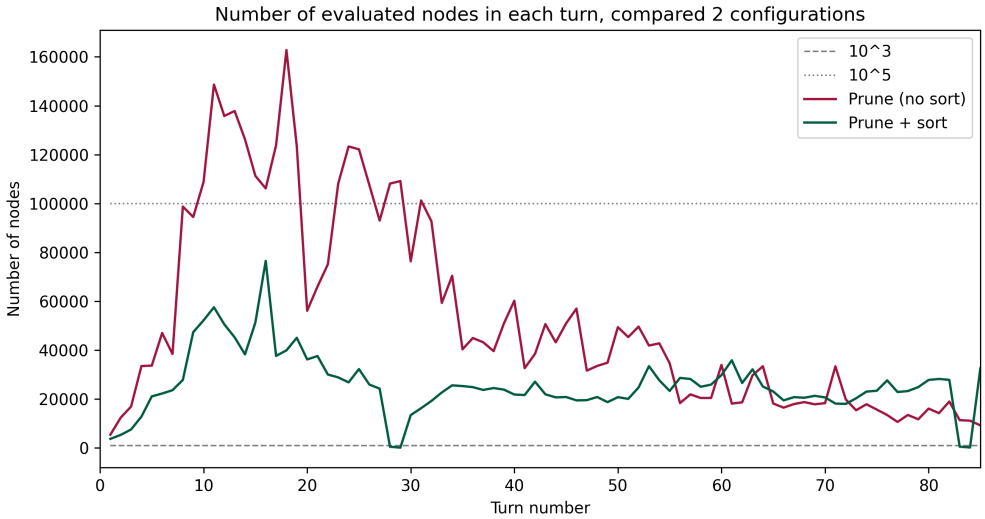
$O(b^d)$ - przypadek pesymistyczny, wartość identyczna jak w przypadku algorytmu Minimax, choć w rzeczywistości wartość ta jest bliższa $O((\frac{b}{log b})^d)$ [5],

$O(\sqrt{b^d})$ - przypadek optymistyczny, gdy najlepsze ruchy przeszukiwane są jako pierwsze.

Aby zbadać w praktyce powyższe zależności przeprowadzono eksperyment polegający na rozegraniu trzech gier - w każdej z nich obaj gracze stosowali identyczne strategie początkowe, środkowe i końcowe, a maksymalna głębokość drzewa była równa 3. Dla każdej z gier zarejestrowano liczbę rozwiniętych/obliczonych węzłów, a także czas potrzebny na wykonanie ruchu w kolejnych turach gry. Różnice między poszczególnymi gramy były następujące: w pierwszej z nich stosowane były zarówno cięcia alfa-beta jak i porządkowanie dzieci węzła, w drugiej jedynie cięcia alfa-beta (bez porządkowania), w trzeciej natomiast nic z powyższych. Otrzymane wykresy zaprezentowane są poniżej.

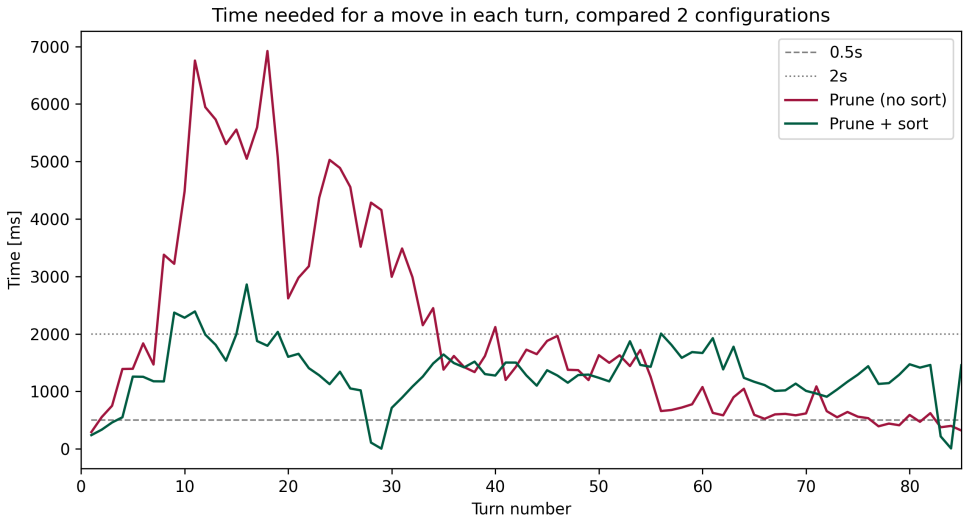


Zdjęcie 4: Porównanie liczby rozwiniętych węzłów dla 3 konfiguracji



Zdjęcie 5: Porównanie liczby rozwiniętych węzłów dla 2 konfiguracji

Na podstawie powyższych wykresów możemy łatwo zauważyć, jak kluczowe w kontekście wydajności programu jest zastosowanie cięć alfa-beta. Wykres ze zdjęcia 4 przedstawia wykładniczy wzrost liczby rozwijanych węzłów dla kolejnych ruchów programu. Nietrudno zauważyć, że już po kilku pierwszych ruchach gry liczba rozwiniętych węzłów w przypadku braku cięć alfa-beta jest rzędy wielkości większa niż dla dwóch pozostałych przypadków (różnica kilku milionów do kilkunastu/kilkudziesięciu tysięcy). Wykres ten szybko się urywa, oczywiście ze względu na trudność sprzętową w przeprowadzeniu symulacji (zasymulowane ok. 16 ruchów zajęło i tak przeszło 12 minut). Warto także spojrzeć na dwa pozostałe wykresy, które przedstawiają fakt, że porządkowanie węzłów także ma istotny wpływ na wydajność algo-



Zdjęcie 6: Porównanie czasu potrzebnego na ruch dla 2 konfiguracji

rytmu. Szczególnie widoczne jest to w pierwszej części rozgrywki, gdzie program niestosujący sortowania musiał przy każdym ruchu rozwinąć nawet kilkukrotnie więcej węzłów, co zajmowało za każdym razem nawet 3-4 sekundy dłużej przy jednym ruchu. Sam fakt, że na końcu lepsze są dla przypadku bez sortowania wynika z tego, że pod koniec gry sortowanie nie musi być już konieczne, ponieważ niesie ono ze sobą pewien narzut obliczeniowy. Jest on jednak rekompensowany z nawiązką we wcześniejszych fazach meczu, gdy drzewo decyzyjne potrafi się bardzo mocno rozrastać.

5 Problemy implementacyjne

Głównym problemem, jaki napotkałem w trakcie implementacji był fakt, że często gra nigdy się nie kończyła - gracze po dojściu w okolice narożnika przeciwnika „błądzili" swoimi pionkami nie potrafiąc znaleźć rozwiązania optymalnego, które byłoby ruchem wygrywającym.

Początkowo testy przeprowadzałem dla głębokości drzewa $d = 2$, ale po kilku próbach rozwiązania tego problemu różnego rodzaju metodami (dopasowanie wag pól, zbiór ruchów tabu, ...) doszedłem do wniosku, że nie ma sensu próbować naprawiać algorytmu dla tak małej wartości d, ponieważ wartość ta oznacza, że drzewo decyzyjne obejmuje pierwszy ruch gracza i odpowiedź przeciwnika. Wymagałoby to prób naprowadzenia w sztuczny sposób gracza na wygrywającą sekwencję, podczas gdy jego horyzont ruchów wynosiłby dokładnie 1. Po zwiększeniu głębokości drzewa do $d = 3$ sytuacja poprawiła się, tzn. czasem program potrafił znaleźć ruchy wygrywające, ponieważ w tej sytuacji mógł planować swoje działania z jednym ruchem do przodu, więc był w stanie zdecydować się na posunięcie, które w kolejnym kroku prowadziło do wygrania partii. Problem polegał jednak na tym, że po zastosowaniu cięć alfa-beta oraz przyjęciu pewnego porządku rozwijania dzieci danego węzła program wciąż potrafił się zawiesić, a gracz będąc o krok od wygrania partii wykonywał naprzemiennie te same ruchy. Okazało się, że gracz potrafił znaleźć ruch, który nie był ruchem wygrywającym, ale ruch następujący po nim (niezależnie od pośredniego ruchu rywala) był już tym kończącym partię. Zapamiętywał więc ten znaleziony ruch, ale tracił przy tym informację o znalezionym ruchu, który powinien być wykonany jako kolejny. Powodem tego było cięcie innych poddrzew, skoro to jedno znalezione miało przypisaną możliwie najlepszą wartość. Następnie, po ruchu przeciwnika, gracz zamiast wybierać ten brakujący ruch, którym zakończyłby partię wybierał (czasem - to zależało od porządku rozwijania węzłów) inny ruch, który ponownie prowadził do zwycięstwa, ale ponownie nie w pierwszym a drugim posunięciu (bo obliczona dla niego wartość była także możliwie najlepsza). Dalej następowały cięcia i nie było już poszukiwania innego ruchu, który także prowadziłby do zwycięstwa będąc 2 poziomy wyżej w drzewie (na głębokości 1 zamiast 3, tzn. sam byłby ruchem wygrywającym).

Początkowo rozwiązanie zakładało wprowadzenie ograniczenia na cięcia tak, aby zostawić algorytmowi „bufor" na znalezienie rozwiązania równie dobrego, ale na płytszym poziomie. Byłoby to jednak obciążające obliczeniowo, ponieważ mimo wszystko większość czasu założenie tego typu nie byłoby ko-

nieczne (problem stanowiła jedynie sama końcówka gry). Ostatecznie więc rozwiązaniem okazało się przeiterowanie po wszystkich dzieciach korzenia drzewa (tzn. wszystkich węzłach wyłącznie na 1. poziomie) pod warunkiem, że rozgrywka jest w fazie końcowej, i sprawdzenie, czy istnieje na tym etapie ruch wygrywający. Ta modyfikacja rzeczywiście rozwiązała problem i to nie tak dużym kosztem, ponieważ na pierwszym poziomie drzewa decyzyjnego jest maksymalnie kilkaset węzłów, choć średnio jest to znacznie mniej (ok. 150-200), więc iteracja po zbiorze tego rozmiaru jest praktycznie niezauważalna, w przeciwieństwie do ograniczenia cięć wykonywanych przez program.

Literatura

[1] Parker Brothers, Rules for the game of Halma
`https://fgbradleys.com/wp-content/uploads/rules/Halma.pdf`. Dostęp: 8.05.2024.

[2] Materiały Politechniki Białostockiej, Problemy z ograniczeniami
`https://www.aragorn.wi.pb.edu.pl/~wkwedlo/EA11.pdf`. Dostęp: 7.05.2024.

[3] Wikipedia, Halma
`https://en.wikipedia.org/wiki/Halma`. Dostęp: 6.05.2024.

[4] wikiHow, How to Win at Chinese Checkers
`https://www.wikihow.com/Win-at-Chinese-Checkers`. Dostęp: 8.05.2024.

[5] Wykład kursu.